Proceeding of the 2004 American Control Conference
Boston, Massachusetts June 30 - July 2, 2004

Matlab-Based Graphical User Interface Development for Basic Stamp 2 Microcontroller Projects

Yan-Fang Li, Saul Harari, Hong Wong, and Vikram Kapila
Mechanical, Aerospace, and Manufacturing Engineering, Polytechnic University, Brooklyn NY 11201

Abstract

Basic Stamp 2 (BS2) is a popular microcontroller used both in hobby and industrial projects. Similar to other microcontrollers, the BS2 programming environment lacks graphical user interface (GUI) capability. In this paper, we present an approach to endow the BS2 microcontroller with GUI capabilities by interfacing it with Matlab and by exploiting Matlab's abundant GUI tools. The proposed Matlab-based GUI environment for BS2 relies on the use of serial communication between the BS2 and a personal computer. We present three examples to demonstrate the efficacy of our approach.

1. Introduction

Microcontrollers are inexpensive devices commonly used in embedded computing applications to impart computing and smart decision-making capabilities to machines, products, and processes. Microcontrollers are designed to interface to and interact with electrical/electronic devices, sensors and actuators, and high-tech gadgets to automate systems. Microcontrollers are generally embedded directly into the product or process for automated decision making. Because they are not meant to interface with human beings, however, microcontrollers do not have graphical user interface (GUI) capabilities that are common in many personal computer (PC) applications. Although this is not a problem for end-user applications, the lack of a GUI is a serious limitation for the developers of microcontroller-based products.

Even though third party GUI tools are available for microcontrollers, they are often inadequate for sophisticated applications and lead to other problems as well. For example, StampPlot [1] is a graphing utility, developed and distributed by Selmaware Solutions, for Basic Stamp 2 (BS2) [2] microcontrollers. This program is fairly easy to use. The user simply connects the BS2 to a PC with a serial cable and gives the StampPlot software control over the serial port on the BS2. The program receives data on the serial port from the BS2. In BS2's programming environment a debug command is used to transmit data serially from BS2 that in turn is plotted by StampPlot. There are three limitations of this software package. First, StampPlot does not provide advanced plotting features, such as three-dimensional plots, which are often of great interest in the development stage of a product. Second, StampPlot cannot co-process data. The program only allows the user to plot the data without the ability to, for example, filter it. Third, we have found that StampPlot interferes with certain PBasic commands. For example, the debugin command, which is used to input a value from the PC to BS2, cannot be used when StampPlot is running.

This work is supported in part by the National Science Foundation under grants 0227479 and 0337668 and the NASA/NY Space Grant Consortium under grant 39555-6519.

In this paper, we address the issue of imparting GUI capabilities to the BS2 microcontroller by combining a powerful software, namely, Matlab, with BS2. Although this paper focuses on BS2 as the microcontroller, our approach can be applied to any microcontroller that supports serial communication, such as the PIC series microcontrollers. This approach can also be extended to where the microcontroller is not being used as a development platform, but rather as a stand alone Data Acquisition and Control Board (DACB). Whereas PC-based DACBs typically cost a few thousand dollars, a PIC microcontroller costs only a few dollars. This application of our approach can be beneficial particularly to educators specializing in the field of control system technology. Finally, this paper is in the spirit of [3], which discusses the integration of BS2 and LabVIEW.

This paper is organized as follows. First, in Section 2, we describe the hardware environment typically used during the development stages of microcontroller-based product design. Next, in Section 3, we describe the software environment used in this paper. In Section 4, we present three examples that illustrate data communication between BS2 and Matlab. Finally, in Section 5, we draw some concluding remarks.

2. Hardware Environment

The hardware environment for this paper consists of a microcontroller, a PC, and a data link between the two. The microcontroller is a device that interfaces to sensors and actuators and performs embedded computing. The PC is used to write the user defined embedded program, which is to be run on the microcontroller. The PC also serves as a debugging environment when prototyping microcontroller based products and processes. It allows the user to receive sensory information and other selected data. A data link is needed for the microcontroller and PC to communicate. In this paper, we use a form of serial communication between the microcontroller and PC.

2.1. Microcontroller

In this paper, we focus on the BS2 microcontroller installed on a Board of Education (BOE) development platform. These devices are manufactured by Parallax, Inc. [4]. The BS2 is a 24-pin Dual Inline Package (DIP) integrated circuit (IC) [2]. It is based on Microchip Inc.'s PIC 16C57 microcontroller. The BS2 is powered by a 6-14V direct current (VDC) power supply. A voltage regulator on the BS2 provides a steady 5VDC supply to the BS2. The BS2 comes with ROM, 2KB Electronically Erasable Programmable ROM (EEPROM), and a small amount of RAM. The BS2 is programmed in PBasic language; the instruction set that is permanently stored on the BS2 ROM. The user-defined program is downloaded into the EEPROM from a PC through a DB-9 serial cable connection between the PC and BOE. The excess EEPROM can be used for long-term data storage. The BS2 has 16

general-purpose digital input/output (I/O) pins that are user defined. The high position on a digital I/O pin refers to a 5VDC and a low position on a digital I/O pin refers to a 0VDC (ground potential). Each pin can source (supply) a maximum current of 20mA and sink (draw) a maximum current of 25mA. The 16 I/O pins on the BS2 at any given time can source/sink a maximum of 40mA/50mA. See [2] for more details on BS2 hardware features.

2.2. Personal Computer

As previously mentioned, the PC is used to write PBasic programs that the BS2 executes and to display sensory data processed by the BS2. Any PC that supports the Stamp Editor and Matlab can be used. In this paper, a Pentium class PC running Windows NT 4.0 is used.

2.3. DB-9 Serial Cable

The BS2 and PC communicate through a serial communication link. A variety of serial communication links are currently in use [5]. The BS2 uses the RS-232 serial communication. The serial cable, which is used in this paper, is called the DB-9 serial cable. The cable links a serial, or COM, port on the PC to the BOE. This allows the user to download a program into the BS2. In addition, this serial connection enables data communication between the BS2 and the PC. The pinout schematic for a DB-9 serial cable is shown in Figure 1. Each of the pins performs a specific task to which it is assigned. The assignment for each pin is shown in Table 1. As noted in [5], only pins 2-5 are used for serial communication between the BS2 and PC.

Table 1: Pin assignments for a DB-9 serial cable

Pin #	Label	Signal Name	Signal Type
1	CD	Carrier detect	Control
2	RD	Received data	Data
3	TD	Transmitted data	Data
4	DTR	Data terminal ready	Control
5	GND	Signal ground	Ground
6	DSR	Data set ready	Control
7	RTS	Request to send	Control
8	CTS	Clear to send	Control
9	RI	Ring indicator	Control

Figure 2 shows the hardware environment used in this paper. A BS2 is shown connected to a PC through a DB-9 serial cable.

3. Software Environment

Serial communication is a low-level protocol used for data communication between two or more devices. As the name implies, serial communication uses a data port to send/receive data in a serial manner, i.e., one bit at a time. Programming two or more devices to communicate serially requires that the devices operate at the same communication rates. As a typical example, a PC and a modem communicate using a serial link. In this paper, a data communication link is established between a BS2 and a PC, where the PC hosts a GUI for the BS2. See [6] for more details on serial communication. Next, we describe various elements of the software environment used in this paper.

3.1. PBasic Program

The BS2 is programmed using the PBasic programming language. It is a Basic-like language developed by Parallax, Inc. for BS2 microcontrollers. In addition to simple arithmetic, the BS2 executes certain task specific commands. We now discuss some commands that are pertinent to our discussion. See [5] for more details on the PBasic programming language.

Debug: debug OutputData

The debug command is of particular importance to our discussion since it relies on serial communication between the BS2 and the PC. The debug command instructs the BS2 to display a string on the PC. The string can be either a simple message, such as *Hello*, or a data string. To display a string of characters, the user places the message that is to be printed to the string within double quotation marks (e.g., debug "Hello World."). The BS2 can display data in one of three formats, binary, decimal, or hexadecimal. The user specifies how the data will be displayed by modifying the debug command with a *bin*, *dec*, or *hex* prefix.

Pulsout: pulsout pin, duration

The pulsout command takes on two arguments, a pin on the BS2, and duration of time in milliseconds. As the name implies it is an output command. Using the pulsout command, the BS2 forces the specified pin high for the duration specified. This command is often used to turn a servomotor to a desired orientation.

Rctime: rctime pin, state, var

The rctime command is a useful function for measuring variations in resistance or capacitance. The function measures the time it takes for a pin to change its state from high to low, or *vice- versa*. The rctime command takes on three arguments, a pin on the BS2, the initial state of the pin, and the variable where the data is stored. The elapsed time is stored in units of two microseconds. When the data exceeds the size of the variable, a null is returned.

Serial communication with a BS2 is accomplished using the following two commands.

Serin: serin Rpin, {\Fpin}, baudmode, {Plabel}, {Timeout, Tlabel}, [InputData]

The serin command is used to import data from another device, the PC in our case, to the BS2. The serin command takes on three to six arguments. The required arguments are Rpin, the pin over which the BS2 will receive data, baudmode, which specifies the speed and configuration at which the devices communicate, and InputData, which instructs the BS2 about where to store the received data. In addition, the serin command can take on an optional Fpin argument, a pin that oversees flow control of the data. The serin command can also take on a Plabel argument, which is used only when the baudmode is set to seven bits and even parity. This argument instructs the BS2 on where it should go in case of a parity error. Finally, the serin command can take on the arguments of Timeout in conjunction with Tlabel. The Timeout argument sets the duration for which the BS2 must wait for incoming data. Tlabel instructs the BS2 on where it must go in the event of a timeout.

Serout: serout Tpin, {\Fpin}, baudmode, {Pace}, {Timeout, Tlabel}, [OutputData]

The serout command is used to export data from the BS2 to another device. The serout command takes on three to six arguments. The three required arguments are Tpin, the pin through which the data will be transmitted, the baudmode which specifies the speed and configuration at which the two devices communicate, and the OutputData, which instructs the BS2 on what data to export. Additionally, the serout command can be modified in one of two ways. One configuration uses the additional arguments of Fpin, a pin

that will monitor the flow of data, and Timeout in conjunction with Tlabel. The Timeout argument instructs the BS2 on how long it must wait for Fpin to give send permission. Tlabel instructs the BS2 on where it must go in the event of a timeout. An alternate configuration uses the Pace argument to set the time between consecutive transmissions of data.

3.2. Matlab Program

Matlab is a commercially available mathematical software package developed and distributed by The Mathworks, Inc. [7]. It is widely used in academia and industry because of its advanced capabilities and a simple user interface. The name Matlab is derived from **Mat**rix **Lab**oratory. Matlab is a powerful software package that allows for plotting data in multiple dimensions. Matlab versions 6.1 and higher support serial communication. Matlab also has built-in toolboxes that contain commonly used engineering functions. We now discuss some commands and toolboxes that are pertinent to our discussion.

Serial: ser_obj = serial(port, 'Baudrate', baud)

The serial command is used to create a serial port object. This command associates an object with a serial port on the PC. The port argument is a string used to specify the port that is to be used to send and receive data. The Baudrate and baud arguments specify the speed at which communication takes place over the serial port. The baud argument is a string referring to the baudrate. If these arguments are omitted, Matlab configures the serial port to communicate at its default settings.

Fopen/fclose: fopen (ser_obj)/fclose (ser_obj)

The fopen and fclose commands are used to connect and disconnect Matlab from the serial port on the PC. The fopen and fclose commands take on a single argument, the object associated with the serial port. The fopen command must be invoked before data can be transmitted over the serial link.

Fscanf/fprintf: data = fscanf(ser_obj, format)/ fprintf (ser_obj, format, cmd, mode)

The fscanf command is used to read data from the device connected on the serial port and to format it. The command takes on two arguments, ser_obj, the object associated with the serial port, and format, the C-language conversion specification that instructs Matlab how to format the data it is reading from the serial port. The data that is imported using this command is stored in the user defined data variable.

The fprintf command is used to write data to the device connected on the serial port. This command takes on four arguments, ser_obj, the object associated with the serial port, format, the C-language conversion specification that instructs Matlab how to format the data it is writing to the serial port, cmd, the string that is being written to the serial port, and mode, the mode of serial communication that Matlab is using. In this paper, we use the asynchronous mode of serial communication, which means data can only be transmitted in one direction at a time.

Freeserial: freeserial(port)

Once Matlab establishes a data link with the serial port, it assumes complete control of the serial port. The freeserial command is used, after closing the port using the fclose command, to force Matlab to relinquish control of the serial port. The command takes on one argument, the port that was used for data communication.

3.3. Simulink

Simulink is MathWorks' icon based programming toolbox [8] that allows the user to construct a block diagram representation of the system being studied. The user can insert block diagrams to specify input functions, system dynamics, controllers, filters, etc., in a Simulink model. The user can similarly specify how and where the output data is to be displayed by Simulink. The Simulink toolbox itself contains many libraries containing functions. The user inserts appropriate function in his/her Simulink block diagram by using a drag-and-drop procedure.

3.4. Dials and gauges library

The dials and gauges library of Simulink allows the user to insert control and display objects into a Simulink model. These objects allow the user to set system parameters and visualize system response within a GUI-based environment.

4. Examples of Serial Communication between the BS2 and Matlab

4.1. PC to BS2 Serial Communication: Servomotor Position Control

This example is used to demonstrate BS2's ability to import data via its serial port. A Parallax standard servomotor (#900-00005) is connected to the BS2. See [9] for instructions on interfacing a servomotor with a BS2. A PBasic program running on the BS2 controls the servomotor position using the pulsout command. In this example, the BS2 receives the duration of the pulsout command from the PC via the serial port. The duration is set within a Simulink program, which exports the pulsout duration to the BS2 via the serial port. Within the Simulink diagram, the range of user input is restricted to ensure safe operation of the servomotor. The user input is entered by turning the needle of the dial shown in Figure 3. The PBasic and Matlab programs corresponding to this example are similar to those given in the Appendix.

4.2. BS2to PC Serial Communication: Plotting Sensor Data

This example is used to demonstrate BS2's ability to export data via its serial port. We begin by connecting a photoresistor to the BS2 in a series RC circuit as shown in Figure 4. Next, we use the rctime function of PBasic to measure the intensity of light as seen by the photoresistor. Finally, we also take advantage of Matlab's ability to co-process the data it receives on the serial port. In particular, we use a low-pass filter between the Matlab function that imports the data and the scope where the data is displayed.

Note that the resistance of the photoresistor is large in dark conditions, on the order of mega-ohms, and small in light conditions, on the order of several hundred ohms. Therefore, when the photoresistor is in dark conditions the time constant of the series RC circuit is large and the rctime command returns a large value. In light conditions, the time constant of the series RC circuit is small, and the rctime command returns a small value. In extremely dark conditions the output of rctime command will exceed the size of a 16-bit variable, thus a null value is returned. For convenience, we do not run this experiment in extreme dark, so that at no time a null is returned by the rctime command.

A PBasic program has been written to continually monitor the light intensity incident upon the photoresistor and to transmit the sensor data using BS2's serial port. Similarly, a Simulink program has been written to import the aforementioned sensory data using the serial port. Once

the data has been imported into the Matlab function, the Simulink program plots it. An experiment was conducted where the light intensity was reduced abruptly at two time instances. As seen in Figure 5(a), raw rctime data is noisy. Thus, in the Simulink block diagram of Figure 6, a low-pass filter has been utilized to provide a smooth data response. This filtered plot of rctime is shown in Figure 5(b). Figure 5 demonstrates the efficacy of using Matlab to co-process data. The PBasic and Matlab programs corresponding to this example are similar to those given in the Appendix.

4.3. Bi-directional Serial Communication between BS2 and PC

This example takes full advantage of the serial communication link between the BS2 and the PC running Matlab. The experimental setup consists of a mechatronics-enabled light reflection experiment that has been developed under a National Science Foundation (NSF) funded Science and Mechatronics Aided Research for Teachers (SMART) program [10] at Polytechnic University. The experiment is designed to demonstrate the law of light reflection. It consists of a light source and a light sensor that can each be rotated independently in the horizontal plane. A common laser pointer is used as the light source. A photoresistor is used as the light sensor. The test bed is built so that the light source and sensor have a common center and axis of rotation. The light source emits light directed at this common center of rotation. Rotation of both light source and sensor is performed using servomotors. The servos direct the light source/sensor to any specific angle using the pulsout command of PBasic. The intensity of the reflected light is measured using a series RC circuit along with the rctime command, where the capacitance value in the series RC circuit is constant.

Figure 7 is an overhead view of the reflection experiment test bed. The left hand of the Figure 7, labeled *incidence*, houses the light source. The right hand side, labeled *reflection*, houses the light sensor.

Figure 8 is a head on, frontal view of the test bed showing the light source/sensor arms. The light source and light sensor each rotate, independently, in the plane coming out of the figure. The light source is located on the left hand side of the figure, and the light sensor is on the right hand side of the figure. In the center of the figure is the experimental material used to reflect the light. The black box at the top of Figures 7 and 8 is the servomotor used to control the rotation of the light sensor. The servomotor used to control the light source is below the test bed, hidden from sight in these two figures.

The experiment was originally designed for the user to set the angle of incidence using the debugin command and to display the rctime data in the debug window on the PC monitor. The BS2 monitors the sensor data and determines the angle of reflection from the position corresponding to the smallest rctime value returned. For this paper, the PBasic program that BS2 executes has been modified to serially receive user command for incidence angle and to serially transmit sensory data. A Simulink program has also been written that allows the user to set the desired angle of incidence using a dial indicator, whose step size has been set to 2˚. This input data is transmitted over the serial link to the BS2, which then sets the angle of incidence accordingly. The BS2 then rotates the light sensor through 90˚ in 2˚ increments. After each 2˚ rotation of the light sensor, the BS2 uses the rctime command to monitor the light intensity as seen by the sensor. This data is then imported into the Simulink program over the serial link. The Simulink-based GUI includes an X-Y plot window in which the imported sensor data is plotted. This allows the user to view the rctime value plotted against the angle of the arm carrying the light sensor. In addition, an on/off switch has been inserted into the Simulink block diagram. The experiment will not run when the switch is in the off position. If the switch is thrown to the off position while the experiment is running, Matlab sends the BS2 a command to stop the experiment and reset the two arms carrying the light sensor and light source to their initial positions. The Simulink block diagram used for serial communication between BS2 and PC is given in Figure 9. The PBasic and Matlab programs corresponding to this example are given in Appendix.

Figures 10(a) and 10(b) are plots of rctime versus the angle of the light sensor. The data in Figure 10(a) was plotted for the case where the angle of incidence was set at 20˚. The angle of reflection corresponds to the angle with highest intensity of light (equivalently, lowest rctime value). Figure 10(b) corresponds to a 40˚ angle of incidence.

5. Conclusion

In this paper, Matlab software and serial communication between a BS2 and a PC have been used to develop a GUI environment for the BS2 microcontroller. We have demonstrated the efficacy of our approach with three examples. In the first example, we used the dials and gauges library of the Simulink toolbox to build a GUI control through which the user supplies parameters to the PBasic program executing on the BS2. In the second example, we used Simulink's advanced computational capabilities to co-process sensor data. Specifically, we used Simulink to filter a noisy signal and to visualize system response within a GUI environment. This example demonstrated the effectiveness of integrating BS2 with Simulink for low-cost data acquisition applications. In the third example, we used bi-directional serial communication between a BS2 and a PC. This example combines elements of the first two examples by allowing the user to input parameters to the PBasic program executing on the BS2 and to visualize system response within a GUI environment. These examples are illustrative of the broad capabilities that the Simulink environment has to offer to augment the capabilities of a microcontroller's programming environment. Whereas commercially available software such as StampPlot provides basic GUI capability for BS2, our approach of imparting advanced GUI capability to microcontrollers using Matlab can be used to develop microcontroller-based low-cost data acquisition and control platforms. In addition, this approach can be used to impart GUI capability to any microcontroller that supports serial communication, such as the PIC series microcontrollers.

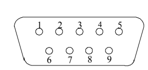


Figure 1: Schematic for a DB-9 serial cable

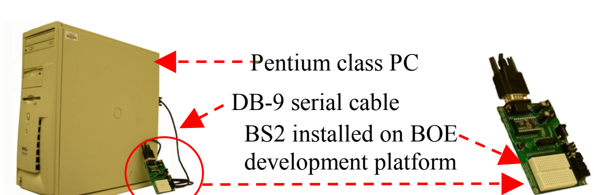


Figure 2: Hardware Environment

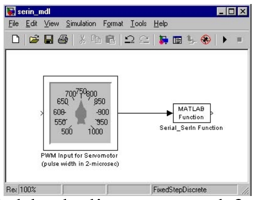


Figure 3: Simulink block diagram used for PC to BS2 serial communication

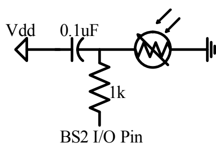
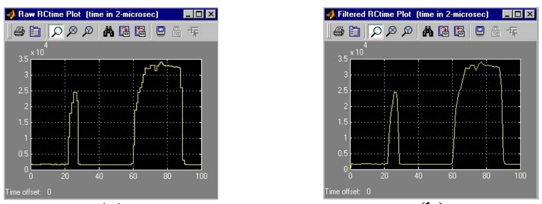


Figure 4: Schematic representation of an RC circuit



(a) (b)
Figure 5: (a) Unfiltered plot of rctime and (b) Filtered plot of rctime

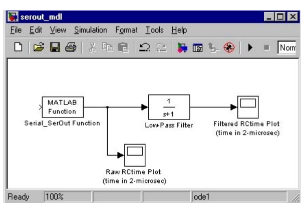


Figure 6: Simulink block diagram used for BS2 to PC serial communication

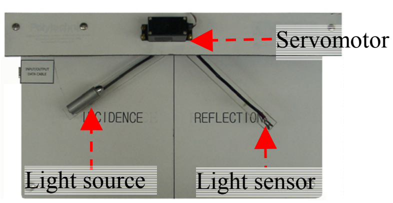


Figure 7: Top view of the light reflection experiment

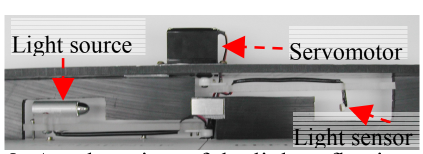


Figure 8: Another view of the light reflection test bed

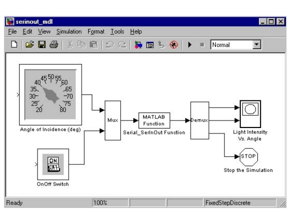
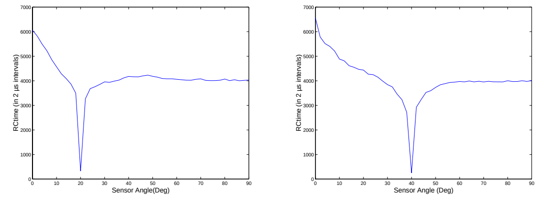


Figure 9: Simulink block diagram used for bi-directional serial communication between BS2 and PC



(a) (b)
Figure 10: Plot of rctime vs. angle of light sensor for (a) 20˚ and (b) 40˚ angles of incidence

References

[1] http://www.selmaware.com, website of Selmaware Solutions developer and distributor of StampPlot.
[2] http://www.parallax.com/detail.asp?product_id=BS2-IC, website of Parallax, Inc. developer and distributor of Basic Stamp 2 microcontroller.
[3] C. J. Radcliffe, "The Basic Stamp II and LabVIEW," http://www.parallax.com/dl/sw/labviewbs2.pdf.
[4] http://www.parallax.com/, website of Parallax, Inc.
[5] *Basic Stamp Programming Manual*, v2.0c, available at http://www.parallax.com/dl/docs/prod/stamps/BasicStampMan.pdf.
[6] K. James, PC Interfacing and Data Acquisition, Newnes, Oxford, U.K. (2000).
[7] http://www.mathworks.com/products/matlab/, website of The Math Works, Inc. developer and distributor of technical computing software Matlab.
[8] http://www.mathworks.com/products/simulink/ website of The Math Works, Inc. developer and distributor of Simulink.
[9] A. Lindsay, *What's a Microcontroller*, v2.0., Parallax, http://www.parallax.com/dl/docs/prod/sic/WAM.pdf.
[10] http://mechatronics.poly.edu, website of Polytechnic's NSF funded Research Experience for Teachers project.

Appendix: PBasic Program and Matlab Function used for Bi-directional Serial Communication between BS2 and PC

1. PBasic Program used for Bi-directional Serial Communication between BS2 and PC

```
'SERIAL_REFLECTION.BS2
'This program was written to read the desired angle of
'incidence from a Simulink function serially. The BS2 sets
'the light source at the desired angle. Next, the BS2 rotates
'the light sensor through 90˚ in two degree increments. The
'BS2 measures the light intensity on the sensor using rctime.
'The BS2 sends the sensor data to the PC serially to be
'plotted in Matlab.
'{$STAMP BS2}
'{$PBASIC 2.5}
'declarartion of constants and variables used
sensor_arm CON 12 'I/O control for the angle of
          'reflection (servo)
source_arm CON 13 'I/O control for the angle of
          'incidence (servo)
sensor_control    CON 5 'I/O control for the light sensor
source_control    CON 6 'I/O control for the light source
sensor_conversion CON 4800/9 'converts angle of
          'incidence into a pulse width command
source_conversion CON 4800/9 'converts angle of
          'reflection into a pulse width command
sensor_offset  CON 490 'pulse width offset for angle of
          'reflection
source_offset  CON 510 'pulse width offset for angle of
          'incidence
```

```
baudrate CON 84 'The BS2 and the PC must be
'configured on the same baud rate to communicate
'successfully on the serial port. In this experiment, the baud
'rate is set at 9600. BS2 automatically converts this
'constant of 84 to a baud rate of 9600 when it executes the
'serin command.
on_off VAR Nib    'var used to store on_off switch value
angle   VAR Word 'var used to store angle (in degrees)
anglein     VAR Word 'var used to store the angle of
            'incidence
anglemax  VAR Word 'var used to store angle of reflection
            '(in degrees)
iter   VAR Word 'var used to move servomotor arm
cycle         VAR Word 'var used to rotate sensor in 2
            'degree increments
source_location  VAR Word 'var used to store angle of
            'incidence (as a pulse width)
light    VAR Word 'var used to store sensor data
max_light VAR Word 'var used to store sensor data of the
            'brightest location
initialize:    'rotates light source and light sensor arms to
            'initial position
low source_control       'turn off light source
FOR iter=0 TO 30
          PULSOUT source_arm,source_offset
          PAUSE 10
          PULSOUT sensor_arm,sensor_offset
          PAUSE 10
NEXT
reflection:
'receive state of on_off switch, if in the on position do the
'following steps a) receive desired angle of incidence b)
'rotate light to that position c) turn on light source if in the
'off position, return to initialize
SERIN 16,baudrate,[DEC on_off]
          IF on_off=0 THEN initialize
SERIN 16,baudrate,[DEC anglein]
max_light = 10000
source_location  =  ((anglein*source_conversion)/100)  +
            source_offset
FOR iter=0 TO 50
          PULSOUT source_arm,source_location
          PAUSE 10
NEXT
HIGH source_control
search:        'search for angle corresponding to highest light
            'intensity
angle=0
anglemax=0
FOR cycle = 0 TO 45
'do the following at 2˚ intervals a) send angle index of
'sensor to the PC b) rotate sensor arm to next angle c) check
'light intensity on the sensor d) send sensor data to the PC e)
'receive state of on_off switch
  SEROUT 16,baudrate,[DEC angle,CR]
  FOR iter=1 TO 30
    PULSOUT  sensor_arm,  angle*sensor_conversion/100
    +sensor_offset
    PAUSE 10
  NEXT
  HIGH sensor_control
  PAUSE 10
  RCTIME sensor_control,1,light
  SEROUT 16,baudrate,[DEC light,CR]
  IF light > max_light THEN cont_check
    max_light = light
    anglemax=angle
cont_check:
IF angle=90 THEN sensor_arm_back
    'if angle=90, rotate the sensor arm to angle of reflection
  SERIN 16,baudrate,[DEC on_off]
  PAUSE 1000
  IF on_off = 0 THEN initialize
    'if switch is in off position, reset the experiment
angle = angle+2
NEXT
sensor_arm_back:
FOR iter=1 TO 100   'rotate the sensor arm to experimental
            'angle of reflection
  PULSOUT sensor_arm, sensor_offset
  +(anglemax*sensor_conversion)/100
          PAUSE 10
NEXT
GOTO reflection
END
```

2. Matlab Function used for Bi-directional Serial Communication between BS2 and PC

```
% The serial_serinout function initializes ser_obj as a serial
% port object. The function sends the position of the on_off
% switch to the BS2 serially. If the switch is in the on
% position, the function receives sensor data, position and
% light intensity, from the BS2, and plots the data on an X-
% Y graph. Matlab continuously monitors the position of
% the on_off switch. If the switch is thrown to the off
% position, the function sends a reset command to the
% BS2 and stops the Simulink simulation.
function y= serial_serinout(u,v)
ser_obj = serial('COM2','baudrate',9600);
ser_obj.terminator = 'CR';
ser_obj.ReadAsyncMode = 'manual';
fopen(ser_obj);
flag=abs(v)
stop = ~flag
fprintf(ser_obj,'%d\n',[flag],'async')
pause(0.5);
if (flag==1)
  angle=u;                      % angle of incidence
  fprintf(ser_obj,'%d\n',[angle],'async')
              % send desired angle of incidence
  A=fscanf(ser_obj,'%d\n')
              % confirm that the BS2 received correct angle
   B=fscanf(ser_obj,'%d\n')
              % receive position of the sensor arm
   C=fscanf(ser_obj,'%d\n')
              % receive light intensity on the sensor
   angle=B;
   rctime=C;
else
          angle=0;
          rctime=0;
end
fclose(ser_obj);
y=[angle;rctime;stop];
```